

# **IF130 Dasar-Dasar Pemrograman**

## **Pertemuan ke 03 – Flowchart Struktur Kendali Pengulangan**

Yustinus Widya Wiratama, S.Kom., M.Sc., OCA

Anak Agung Ngurah Ananda Kusuma, BE(Hons), MEng, PhD

Putri Sanggabuana Setiawan, S.Kom., M.T.I.

# Capaian Pembelajaran Mingguan Mata Kuliah (Sub-CPMK):



1. **Sub-CPMK 04:** Mahasiswa mampu menggambar flowchart dengan struktur kendali pengulangan (C3)

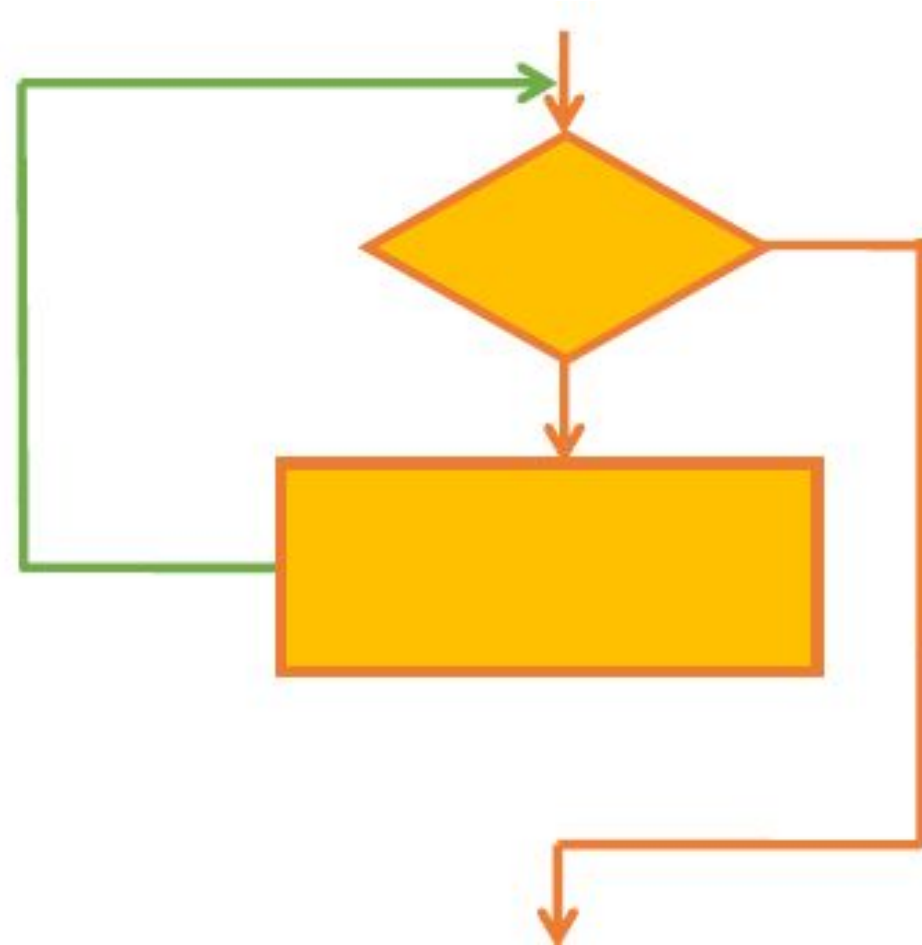
# Review

- Algorithm can be defined as a set of *detailed, unambiguous* and *ordered instructions* developed to describe the **processes** necessary to produce the desired **output** from a given **input**.
- Flowchart and pseudocode are both popular ways of representing algorithms.
- There are 3 variations of **selection control structure**:
  - Simple IF statement
  - Null ELSE statement
  - Nested IF statement
- The **case structure** is another way of expressing a nested IF statement.

# Outline

1. Definition of repetition control structure
2. Kind of repetition control structure
3. Flowchart of repetition control structure
4. Exercises

# DEFINITION & KIND OF REPETITION



# Repetition Essentials

- A loop is a **group of instructions** the computer executes repeatedly while some loop repetition condition remains true.
- 2 kinds of repetition
  1. Sentinel-controlled repetition
  2. Counter-controlled repetition

# Sentinel-Controlled Repetition

- Sentinel-controlled repetition is sometimes called **indefinite repetition** because it's not known in advance how many times the loop will be executed
- Sentinel values are used to control repetition when
  - The precise number of repetitions is not known in advance
  - The loop includes statements that obtain data each time the loop is performed

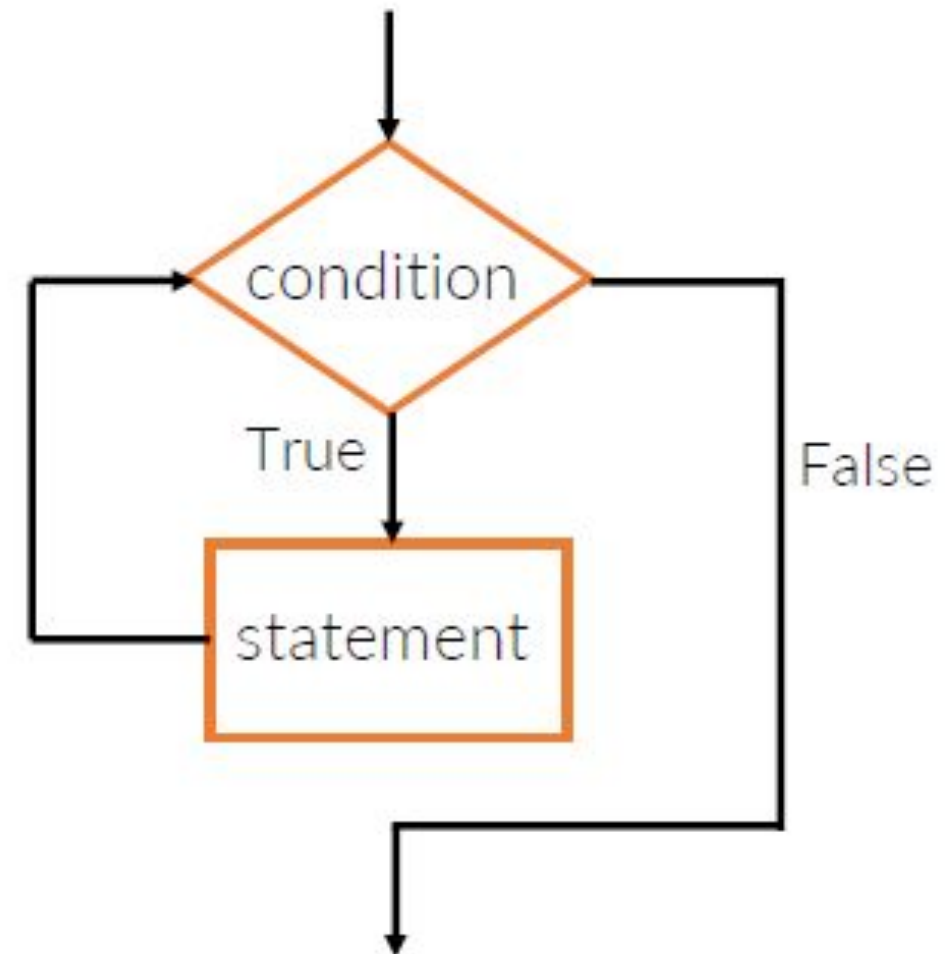
# Sentinel-Controlled Repetition

- **While**
- **Do-While**
- **Do-Until**
- Both the **While** and **Do-While** loops cause a statement or set of statements to repeat **as long as** a condition is **true**.
- The **Do-Until** loop causes a statement or set of statements to repeat **until** a condition is **true**.



# WHILE Loop

- “While a condition is true, do some task.”
- The loop is repeated when the condition is true (when its value is not 0).
- The loop is exited when the condition is false.

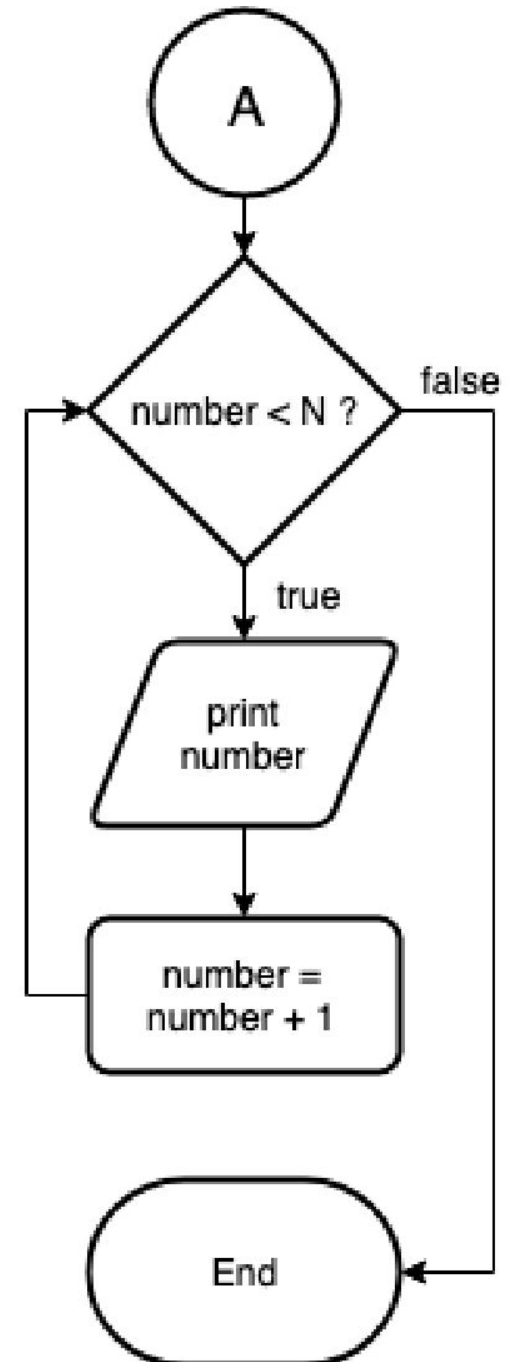
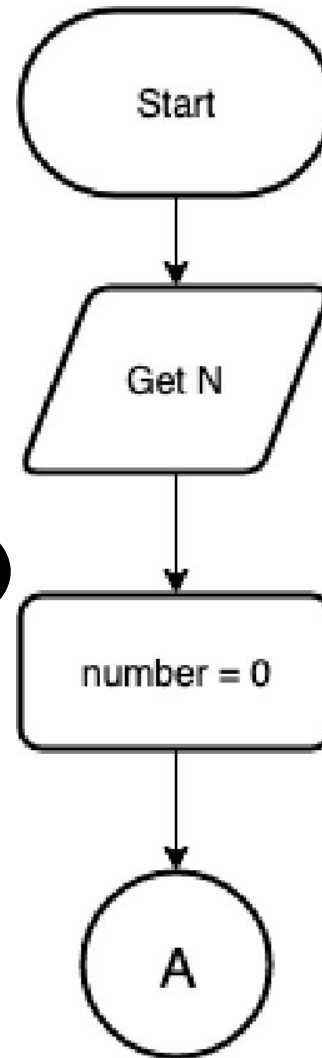


# WHILE Loop

Program output  
(with Input Shown in **Bold**)

**10 [enter]**

0 1 2 3 4 5 6 7 8 9



# WHILE Loop

## Program Output (with Input Shown in Bold)

Enter the amount of sales.

**10000.00** [Enter]

The commission is \$1000

Do you want to calculate another commission? (Enter y for yes.)

**y** [Enter]

Enter the amount of sales.

**5000.00** [Enter]

The commission is \$500

Do you want to calculate another commission? (Enter y for yes.)

**y** [Enter]

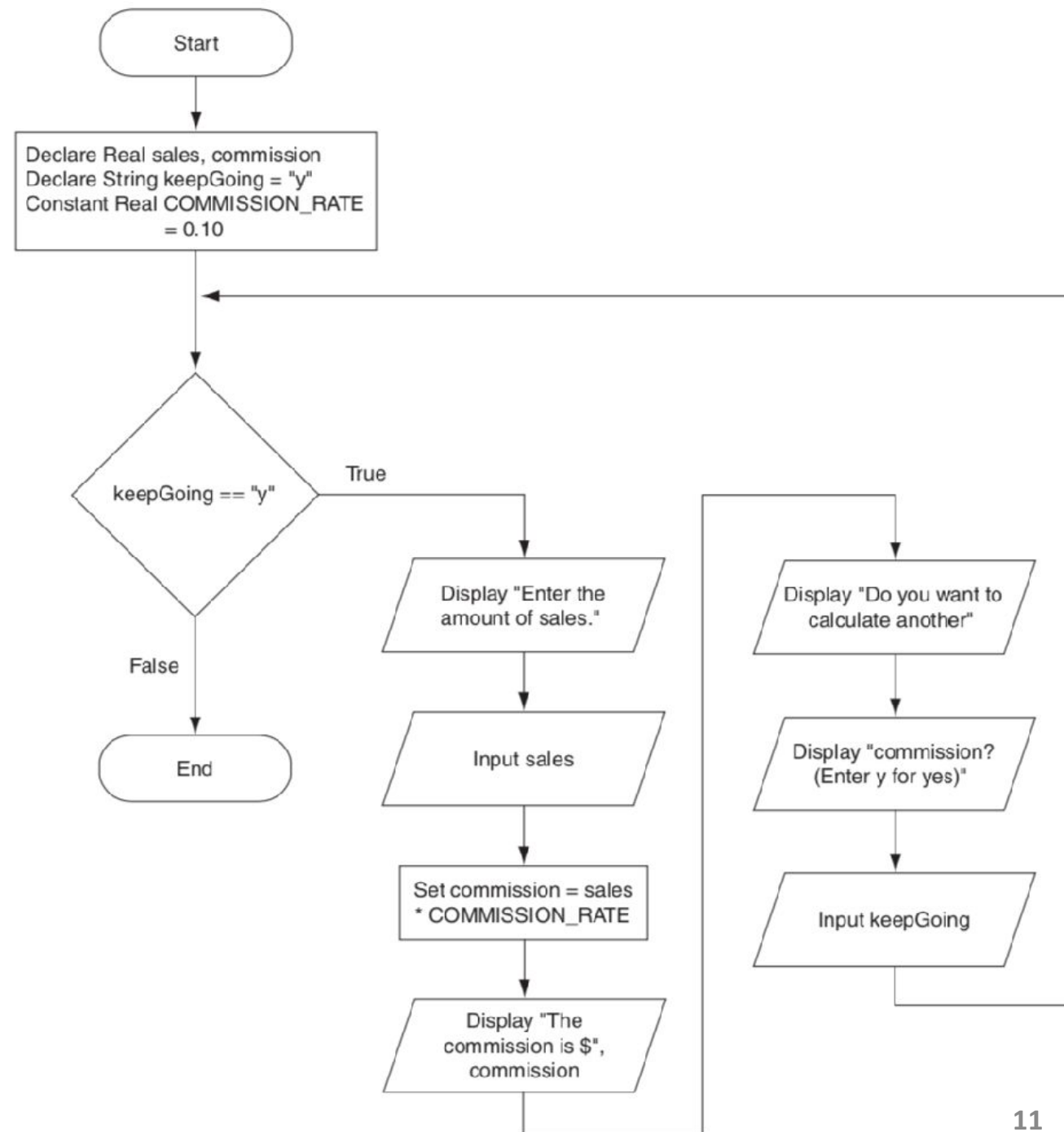
Enter the amount of sales.

**12000.00** [Enter]

The commission is \$1200

Do you want to calculate another commission? (Enter y for yes.)

**n** [Enter]



# WHILE Loop

## Program Output (with Input Shown in Bold)

Enter the substance's temperature.

**104.7** [Enter]

The temperature is too high.

Turn the thermostat down and wait five minutes. Take the temperature again and enter it here.

**103.2** [Enter]

The temperature is too high.

Turn the thermostat down and wait five minutes. Take the temperature again and enter it here.

**102.1** [Enter]

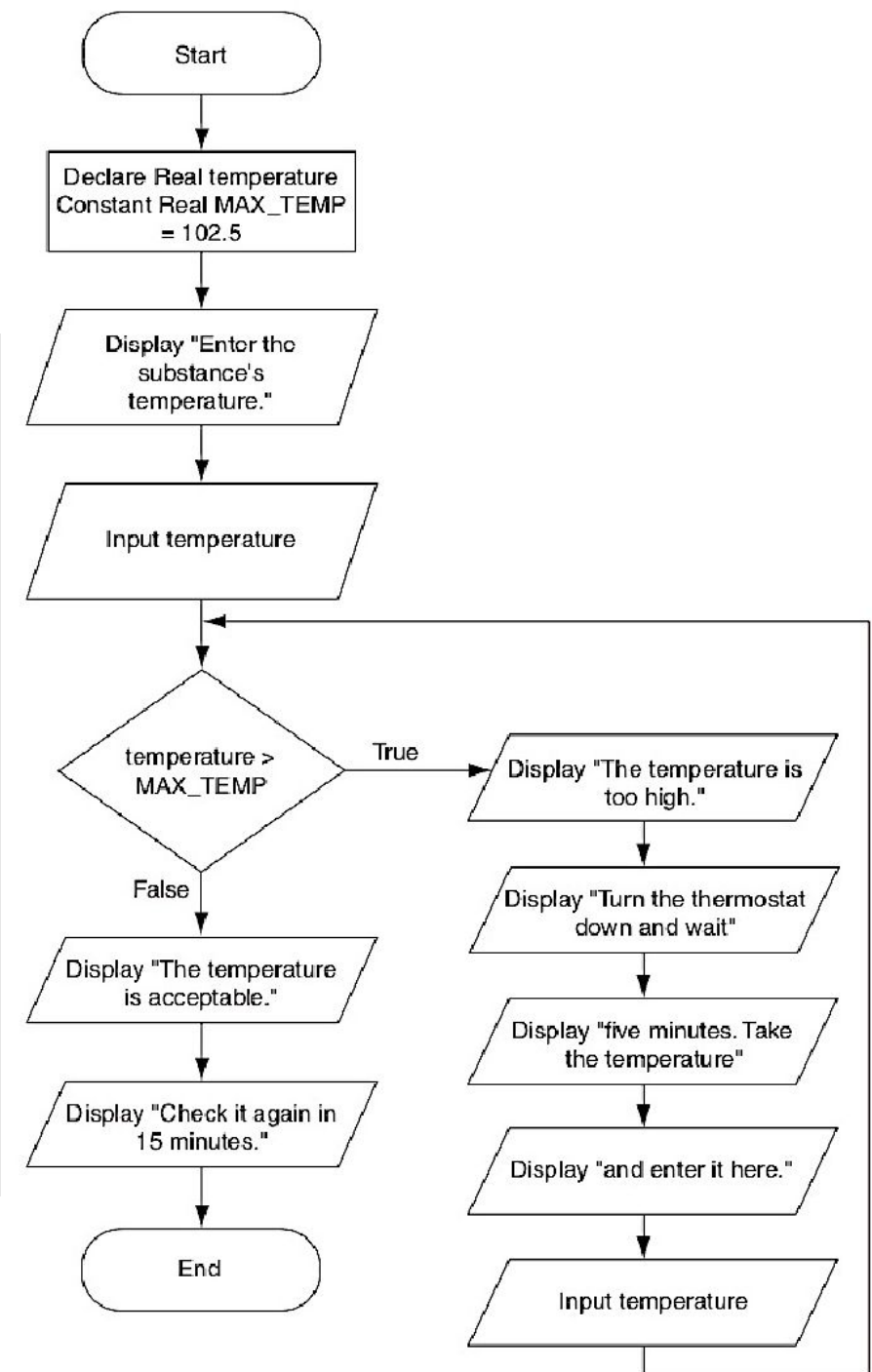
The temperature is acceptable.  
Check it again in 15 minutes.

## Program Output (with Input Shown in Bold)

Enter the substance's temperature.

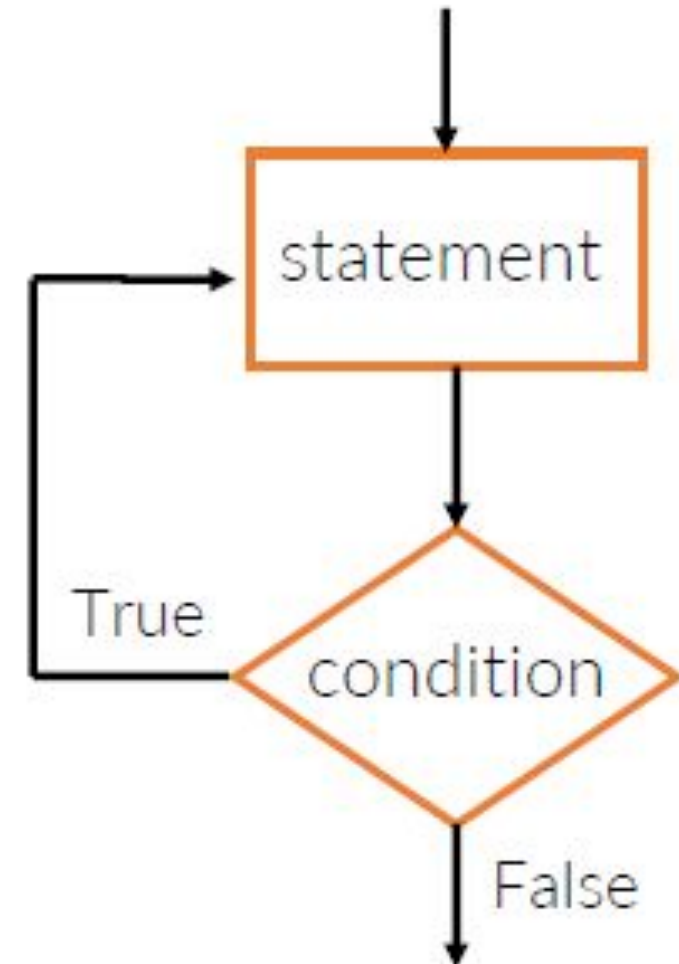
**102.1** [Enter]

The temperature is acceptable.  
Check it again in 15 minutes.



# DO-WHILE Loop

- The **Do-While** loop is a **posttest** loop. This means it performs an iteration before testing its condition.
- As a result, the Do-While loop **always performs at least one iteration**, even if its condition is false to begin with.

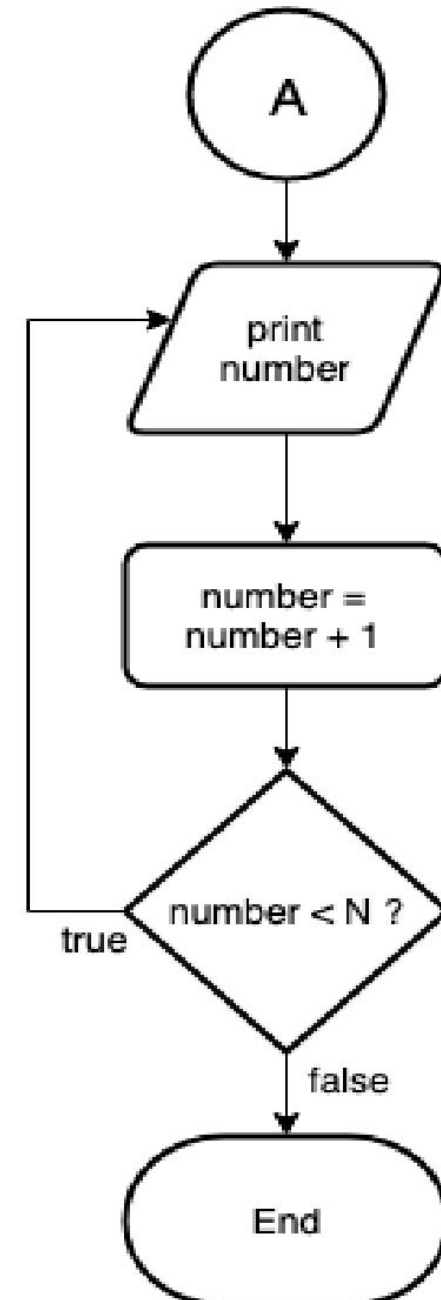
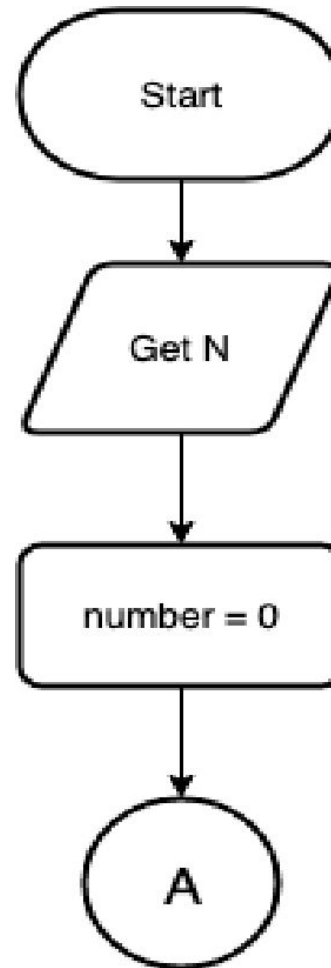


# DO-WHILE Loop

Program output  
(with Input Shown in **Bold**)

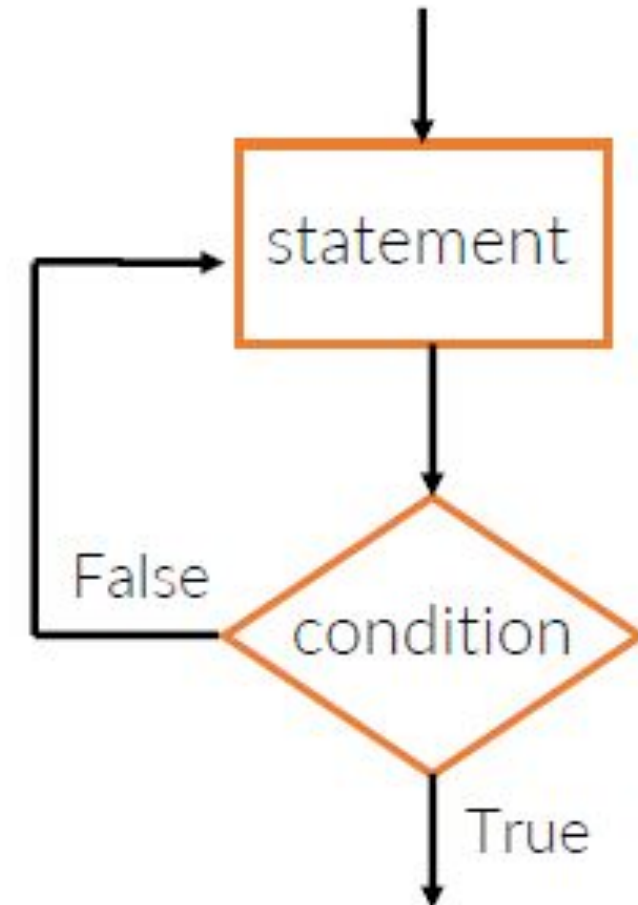
**10 [enter]**

0 1 2 3 4 5 6 7 8 9



# DO-UNTIL Loop

- Sometimes, however, it is more convenient to write a loop that iterates **until** a condition is **true**—that is, a loop that iterates as long as a condition is false, and then stops when the condition becomes true .

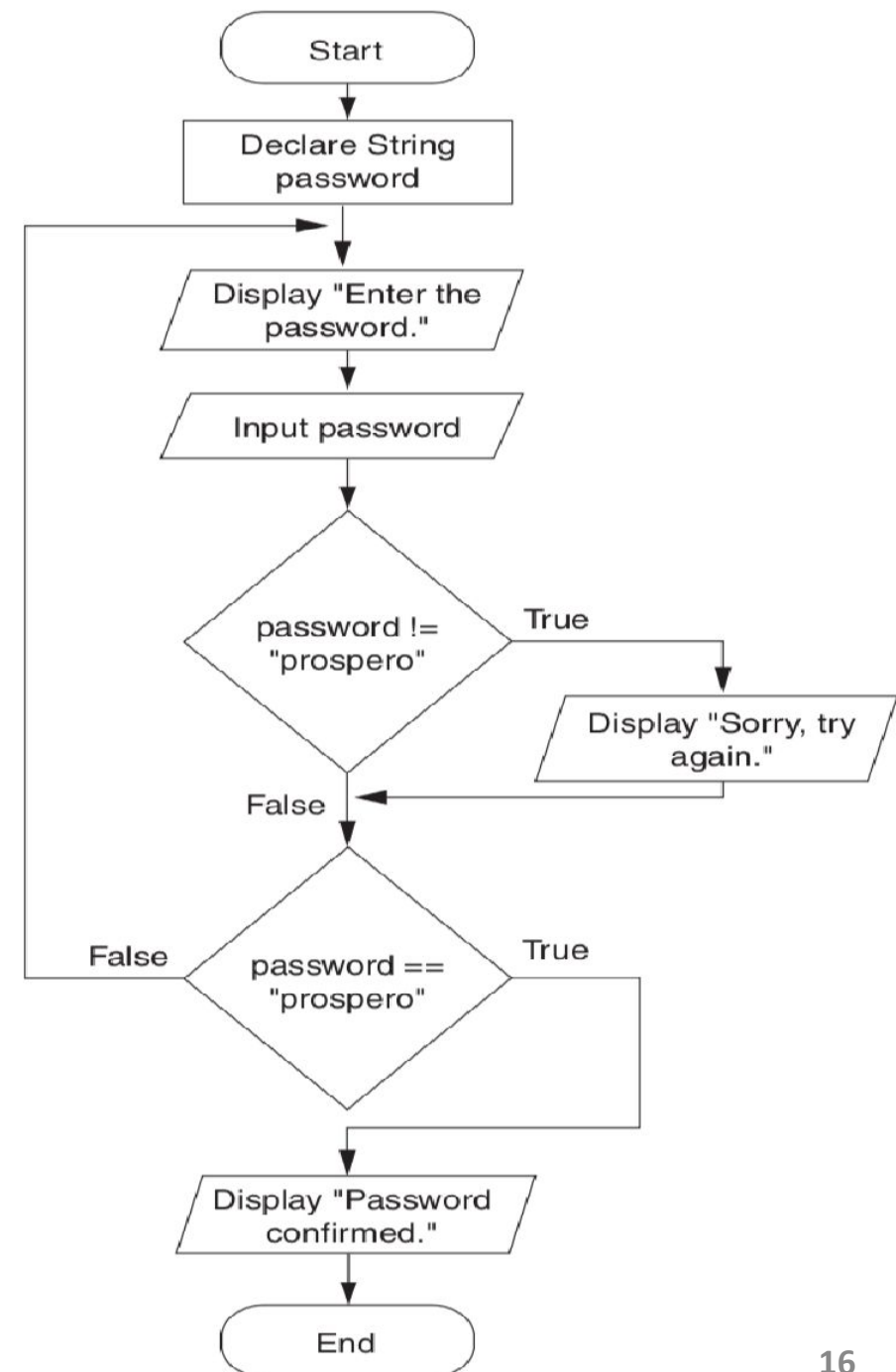




# DO-UNTIL Loop

## Program Output (with Input Shown in Bold)

```
Enter the password.  
ariel [Enter]  
Sorry, try again.  
Enter the password.  
caliban [Enter]  
Sorry, try again.  
Enter the password.  
prospero [Enter]  
Password confirmed.
```

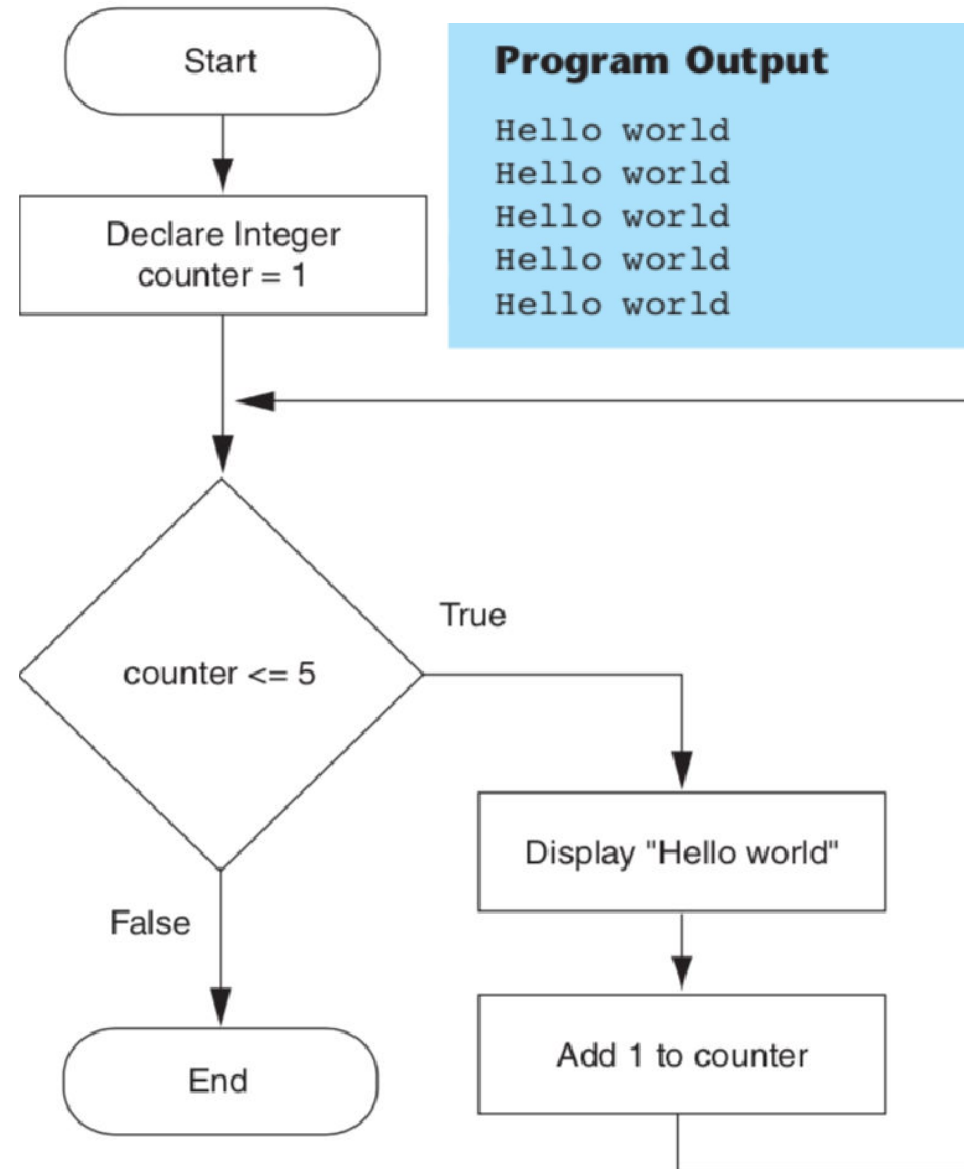
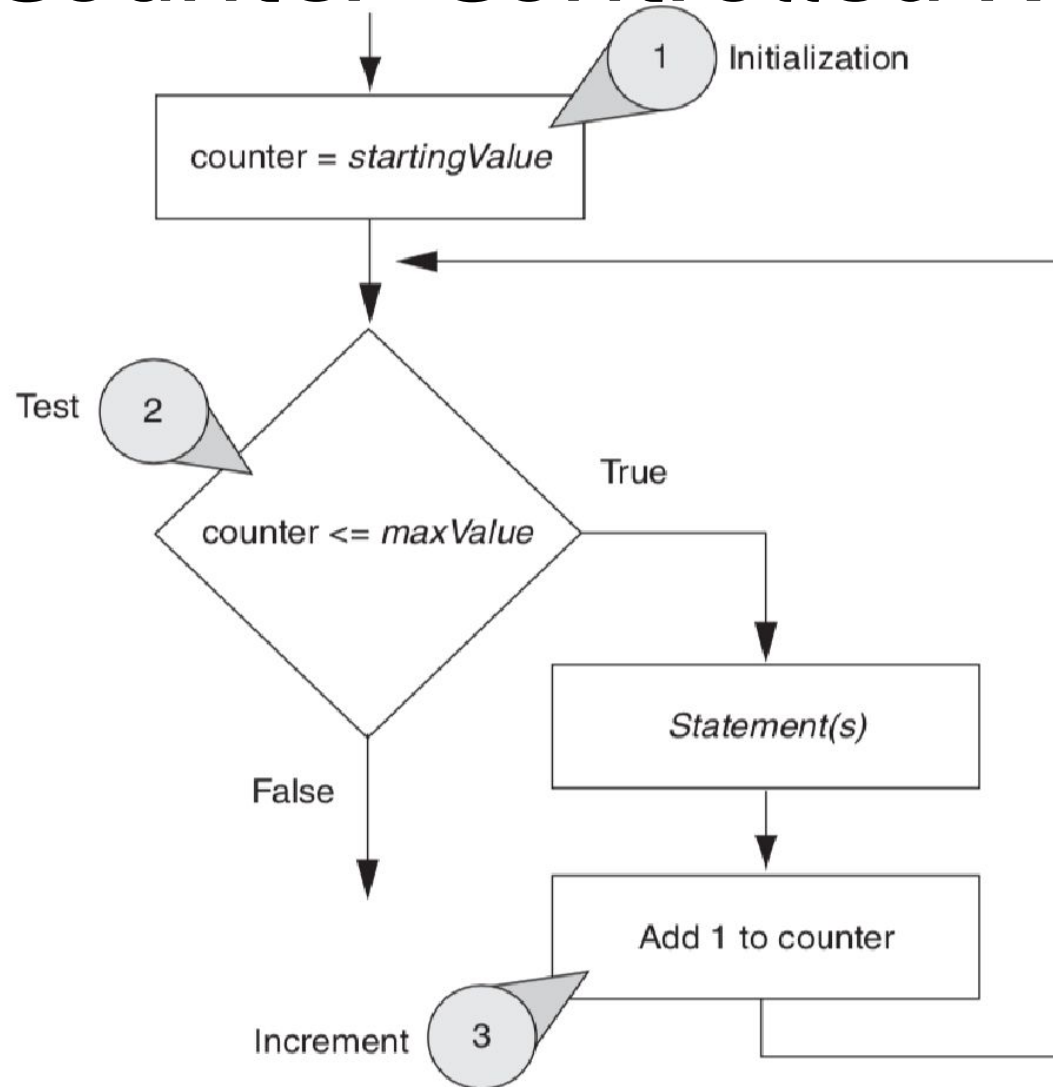




# Counter-Controlled Repetition

- Counter-controlled repetition is sometimes called **definite repetition** because we know in advance exactly how many times the loop will be executed.
- This is usually called the **For statement**.
- A loop control variable is used to count the number of repetitions
  1. **Initialization:** Loop control variable is set to an initial value before the while statement is reached
  2. **Testing:** Loop control variable is tested before the start of each loop repetition
  3. **Updating/Increment:** Loop control variable is updated (incremented / decremented) during each iteration

# Counter-Controlled Repetition



## Program Output

```
Hello world
Hello world
Hello world
Hello world
Hello world
```

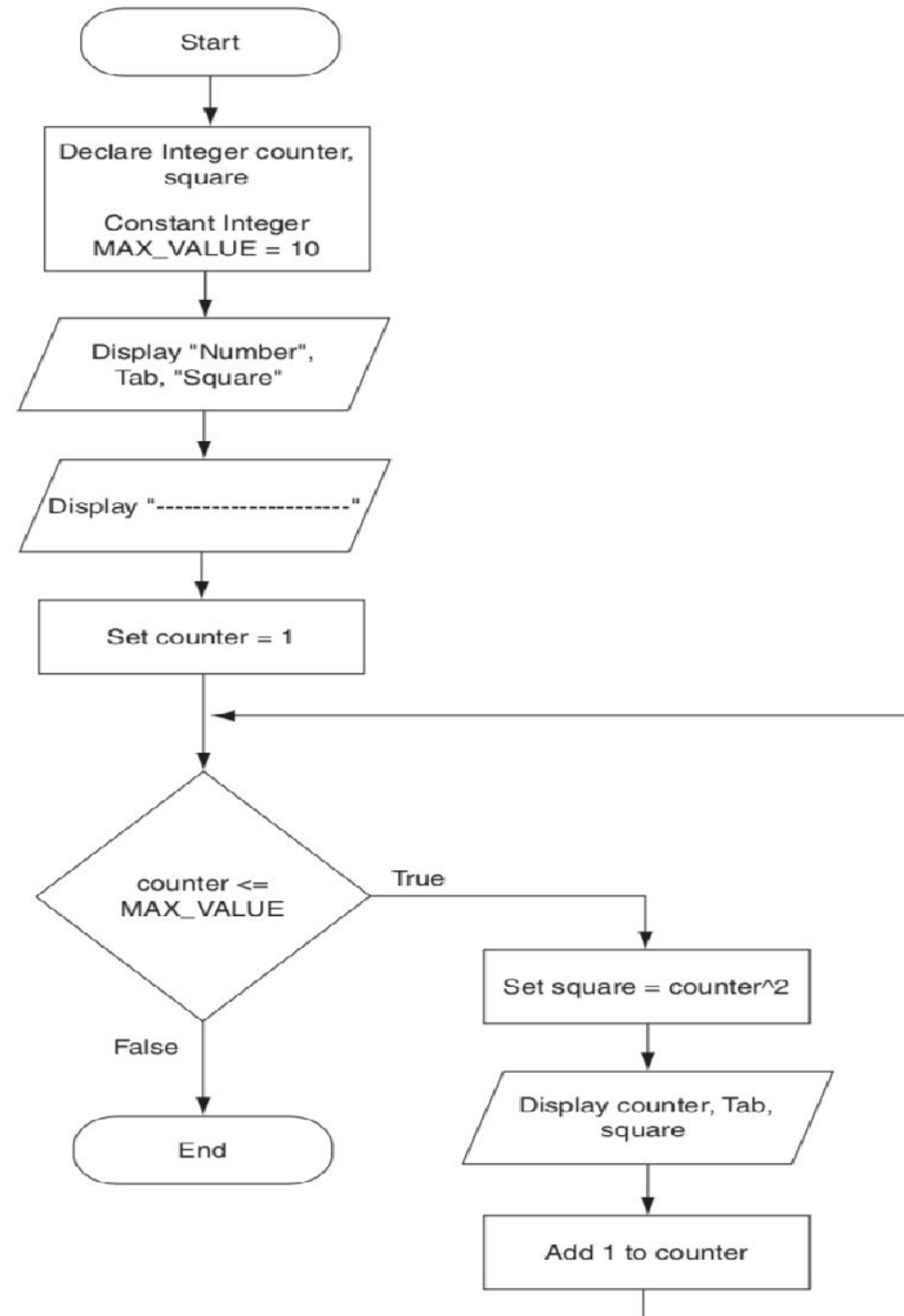
# FOR Statement

- In some situations, it is also helpful to use the counter variable in a calculation or other task within the body of the loop.
- For example, suppose you need to write a program that displays the numbers 1 through 10 and their squares.

# FOR Statement

## Program Output

Number	Square
-----	
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100



# Practice 1

- Design an algorithm in **flowchart** which displays the numbers 1 through the maximum value (user input) and their squares.
- Maximum value = 5

Number	Square
1	1
2	4
3	9
4	16
5	25

# Practice 2

- Design an algorithm in **flowchart** that print the following sequence of values

20	14	8	2	-4	-10
----	----	---	---	----	-----

# Practice 3

- Design an algorithm in **flowchart** that print the following sequence of values

19	27	34	40	45
----	----	----	----	----

# Practice 4

- The factorial function is used frequently in probability problems. The factorial of a positive integer  $n$  (written  $n!$  and pronounced “ $n$  factorial”) is equal to the product of the positive integers from 1 to  $n$ . Draw a **flowchart** that evaluates the factorials of the integers from  $p$  to  $q$  ( $p$  and  $q$  are inputted by user). The screen dialogue should appear as follows:

<u>1</u>	<u>5</u>	
1!	=	1
2!	=	2
3!	=	6
4!	=	24
5!	=	120

<u>3</u>	<u>8</u>	
3!	=	6
4!	=	24
5!	=	120
6!	=	720
7!	=	5040
8!	=	40320



# NEXT WEEK'S OUTLINE

1. Pseudocode of repetition control structure
2. Desk checking
3. Exercises

# REFERENCES

- Gaddis, Tony, 2019, Starting out with programming logic & design, Fifth edition, Pearson Education, Inc.

# Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.



# Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.